Preprocessing Algorithms for Scalable Quantum Annealing

Team: Hristo Djidjev (CCS-3)

Georg Hahn (Imperial College, UK)

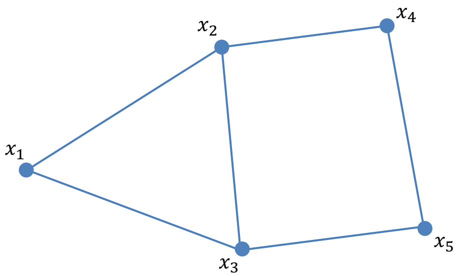
Guillaume Rizk* (INRIA-Rennes, France)

April 27, 2017

LA-UR-17-23726

Background and motivation

- LANL's D-Wave has a relatively small number (1095) of qubits
  - Problem sizes restricted to ~1000 variables
- Only a small fraction of these qubits are typically used because of the penalties
  - Example: Max Clique



Constraint: $$\sum_{i=1}^{n} x_i = s$$

Penalty: $$M\left(\sum_{i=1}^{n} x_i - s\right)^2 = M\left(\left(\sum_{i=1}^{n} x_i\right)^2 - 2s\sum_{i=1}^{n} x_i + s^2\right)$$

  - Results in a dense QUBO matrix, regardless of input graph

Solving bigger problems

- Because of dense QUBOs, sizes fitting DW even smaller
  - Chimera can embed complete graphs of ~45 vertices
  - More than 95% of qubits used for connections
- Can use decomposition to solve bigger problems, but
  - Issue: # subproblems may grow as $\exp\left(\frac{\text{prob\_size}}{\text{subprob\_size}}\right)$
  - No quantum advantage if $\text{subprob\_size} \leq 300$
- Fit-size for dense problems grows only as $\sqrt{\#qubits}$
  - Hardware upgrades will not resolve issue soon
- The solution: increase the size of problems directly fitting D-Wave

Objectives

- Develop methods that allow larger problems to fit into D-Wave
- Work on level of QUBO matrix
  - Hence problem independent
  - Same method can be used for solving different problems
- Two approaches:
  - Remove entire rows and columns from the QUBO matrix
  - Remove (set to zero) individual elements of the matrix
- Second approach (not discussed today):
  - Use spectral sparsification theory
  - Guarantees that resulting matrix approximates the original one within a user specified accuracy

Roof duality and persistency

- Roof duality
  - Technique for computing lower bounds for quadratic forms
  - Based on theoretical work from 1980s
  - Recently used in computer vision
  - Converts quadratic form into quadratic *posiform*
    - Posiform example: $f(x_1, x_2, x_3) = -2 + 0.5\bar{x}_2 + \bar{x}_1 x_2 + x_2 x_3 + 2\bar{x}_1\bar{x}_3$
  - Posiform analysis can be used to deduce *persistencies*
- Persistency
  - Strong/weak persistency: valid for all/some optimal assignments
    - Example strong: $x_2 = 0, x_7 = 1$ for <u>all</u> optimal assignments
    - Example weak: $x_3 = 1, x_5 = 0$ in <u>some</u> optimal assignment

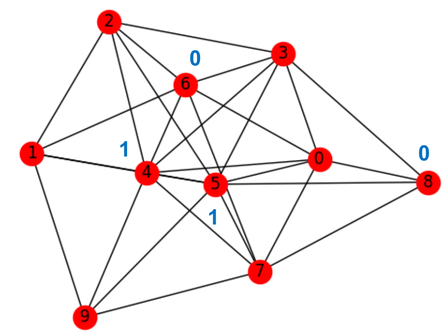
Discovering persistencies

- Algorithm outline
  i. Convert QUBO matrix into a posiform
  ii. Convert posiform into a graph
  iii. Solve maxflow problem on graph
  iv. Analyze results to discover persistencies

- Implementation: adapted software from
  – QPBO (C.Rother, V. Kolmogorov, V. Lempitsky, M. Szummer)
  – pyqpbo (A. Mueller)

Illustration of method

- Input graph
  - Find a maximum clique



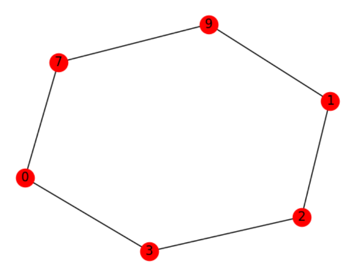
- Construct QUBO

-1	2	2	0	0	0	0	0	0	2
2	-1	0	2	0	0	0	2	2	0
2	0	-1	0	0	0	0	2	2	2
0	2	0	-1	0	0	0	2	0	2
0	0	0	0	-1	0	0	0	2	0
0	0	0	0	0	-1	2	0	0	0
0	0	0	0	0	2	-1	0	2	2
0	2	2	2	0	0	0	-1	0	0
0	2	2	0	2	0	2	0	-1	2
2	0	2	2	0	0	2	0	2	-1

- Analyze for persistencies

  [-1 -1 -1 -1 1 1 0 -1 0 -1]

- Simplify problem



Experimental setup

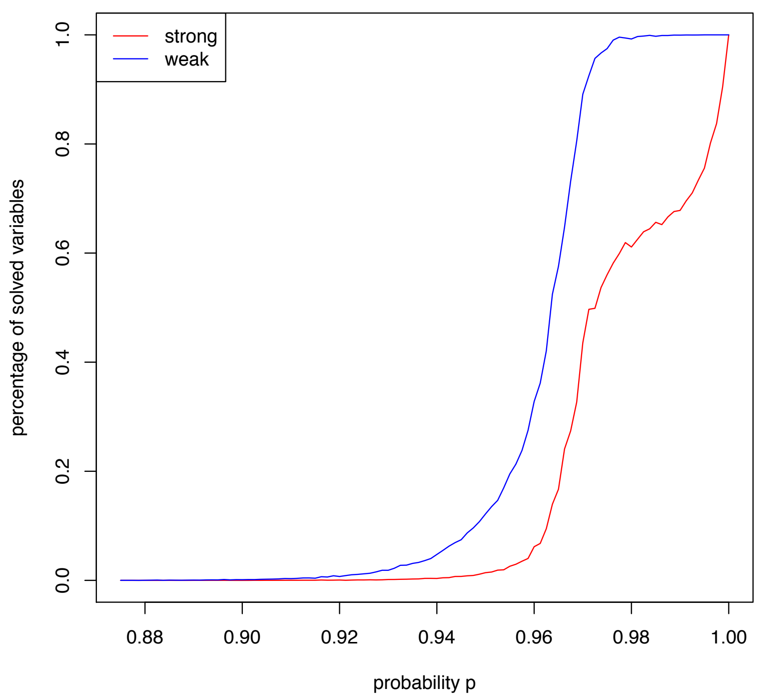
- Goals: determine:
  - What features affect method's effectiveness
  - If combining with decomposition methods has synergetic effect
  - If problem formulation matters
- Optimization problems
  - Maximum Clique
  - Maximum Cut
- Test instances
  - C-fat rings – regular, sparse
  - Hamming graphs – regular, dense
  - Random – no structure
  - Geometric – geometric structure

Persistencies for Max Clique

Name	Vertices/ Variables	Edges	QUBO density	Clique size	Persistencies
C_FAT_200_1	200	1534	92.29%	12	100%
C_FAT_200_5	200	8473	57.42%	58	100%
C_FAT_500_1	500	4459	96.43%	14	0%
C_FAT_500_5	500	23191	81.47%	64	0%
HAM_6_2	64	1824	9.52%	32	100%
HAM_6_4	64	704	65.08%	4	0%
HAM_8_2	256	31616	3,14%	128	100%
HAM_8_4	256	20864	36.08%	16	0%

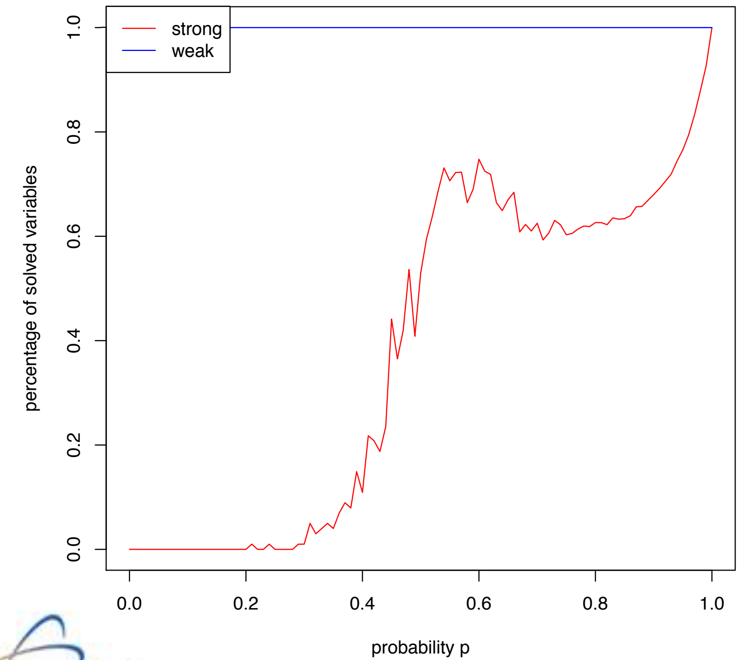
Adding random edges

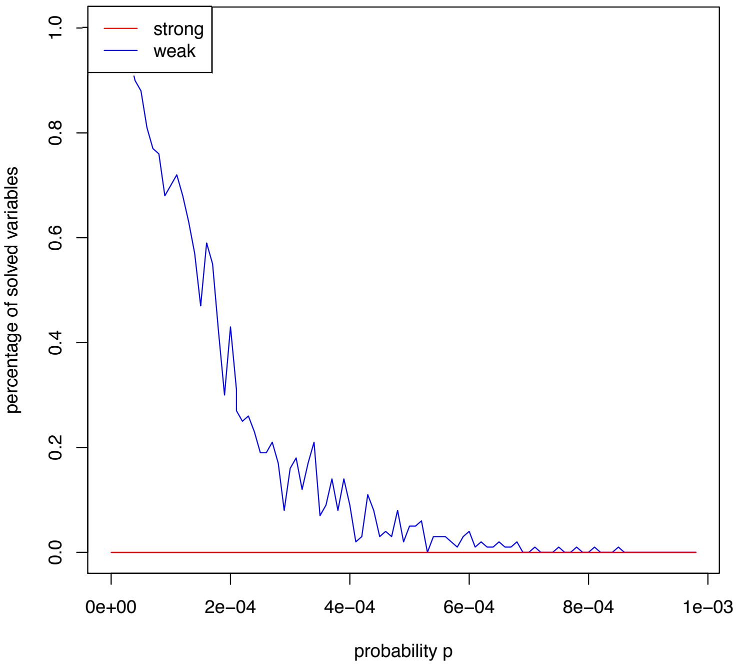
- Start with a graph with no persistencies
- Add increasing number of random edges
- See how the # persistencies change



Removing random edges

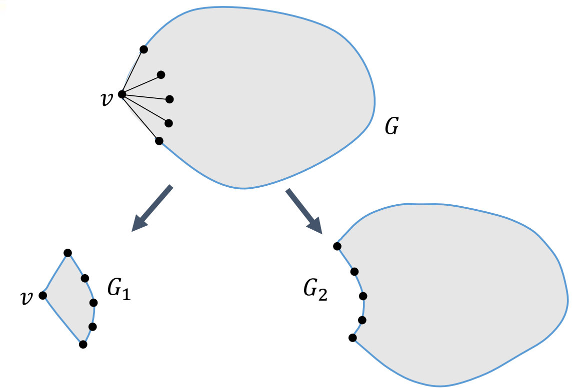
- Start with a graph with 100% weak persistencies
- Add/remove increasing number of random edges
- See how the # persistencies change



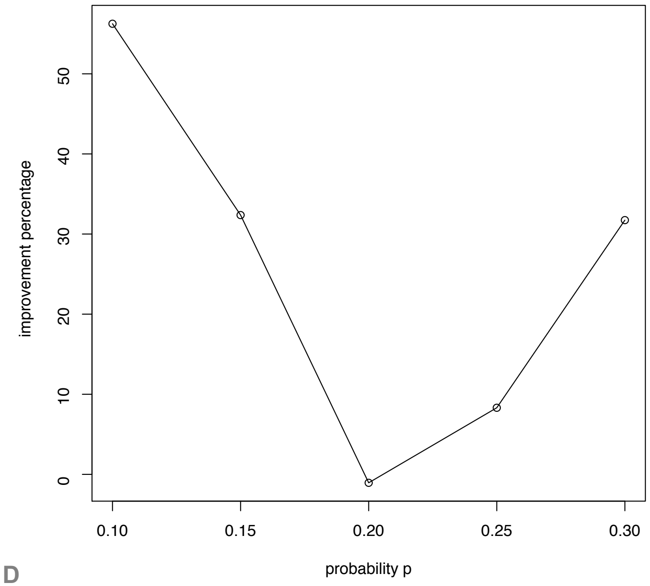


Combining with decomposition algorithms

- Use the the most general of the algorithms that removes one vertex at each iteration



- Combine with persistency algorithm
  - Upto 60% reduction in number of subproblems
  - Probably could do even better



Comparing different formulations

- Do formulations matter for # of persistent variables?
- If they do, one can look for more favorable ones
- Maximum clique problem
  - "independent set" formulation

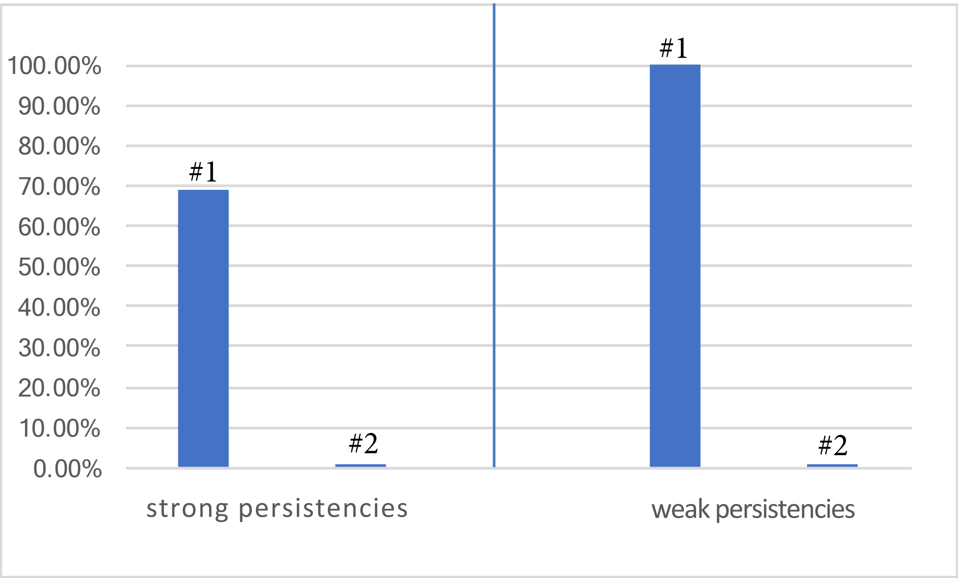
$$H = -\sum_{v \in V} x_v + 2 \sum_{(u,v) \in \overline{E}} x_u x_v,$$

  - "edge-counting" formulation, assumes MC size K is known

$$H_K = (K+1)\left(K - \sum_{v \in V} x_v\right)^2 + \left[\frac{K(K-1)}{2} - \sum_{(u,v) \in E} x_u x_v\right]$$

Results

- Comparison of the two formulations
- Graphs used are from the C-fat family



Maximum cut problem

- The problem
  - The vertices of the graph have to be divided into two sets
  - The cut is the set of cross edges
  - The size or the weight of the cut has to be maximized
  - Equivalent to the minimum cut problem with real weights
- D-Wave formulation
  - Ising

$$Is(x) = \sum_{(uv)\in E} x_u x_v , \quad x_u \in \{-1, 1\}$$

  - QUBO

$$Q(x) = \sum_{(uv)\in E} \left(x_u(1-x_v) + (1-x_u)x_v \right) , \quad x_u \in \{0, 1\}$$

Experimental results for Max Cut

	n	p	persistencies
R graphs (random)	500	2.50	13.40
	500	5.00	100.00
	1000	2.50	11.40
	1000	5.00	100.00
G graphs (geometric)	500	5.00	1.60
	500	10.00	0.40
	1000	5.00	2.70
	1000	10.00	0.00

Conclusions

- Need ability to fit larger problems into D-Wave in order to see a quantum advantage any time soon
- Persistency-based methods
  - Good candidates to reduce the sizes of QUBOs
  - General methods, can be applied to any problem
  - Early results, much more work needed
- Performance varies significantly even between very similar problems
- Combination with decomposition methods can reduce the number of problems by upto 60%
- Choosing the right formulation can have huge impact on effectiveness

Future work

- Adapt algorithms to better work for <u>combinatorial</u> problems
  - Current implementations target computer vision applications
- Exploit knowledge of the <u>particular</u> optimization problem solved
  - Currently information only from QUBO matrix used
- <u>Characterize</u> problems/formulations/inputs for which the method works better
  - Most optimization problems have multiple formulations
- <u>Combine</u> with other methods to increase effectiveness
  - Small changes to matrix can result in many new persistencies